

# DS 621 (Fall 2020): Homework 2 (Group3)

## Classification Metrics

Zach Alexander, Sam Bellows, Donny Lofland, Joshua Registe, Neil Shah, Aaron Zalki

Source code: [https://github.com/djlofland/DS621\\_F2020\\_Group3/tree/master/Homework\\_2](https://github.com/djlofland/DS621_F2020_Group3/tree/master/Homework_2)

### Overview

This assignment will present various classification metrics through creating functions in R that will carry out these calculations. These calculations will be compared against built-in functions from various R packages and a graphical representation of these results will be presented.

### 1. Download Dataset

```
df <- read.csv ("https://raw.githubusercontent.com/djlofland/DS621_F2020_Group3/master/Homework_2/datas
```

The dataset has three key columns we will use:

`class` the actual class for the observation

`scored.class` the predicted class for the observation (based on a threshold of 0.5)

`scored.probability` the predicted probability of success for the observation

### Data Exploration

```
summary(df)
```

```
##      pregnant      glucose      diastolic      skinfold
##  Min.   : 0.000   Min.   : 57.0   Min.   : 38.0   Min.   : 0.0
## 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.0   1st Qu.: 0.0
## Median : 3.000   Median :112.0   Median : 70.0   Median :22.0
## Mean   : 3.862   Mean   :118.3   Mean   : 71.7   Mean   :19.8
## 3rd Qu.: 6.000   3rd Qu.:136.0   3rd Qu.: 78.0   3rd Qu.:32.0
## Max.   :15.000   Max.   :197.0   Max.   :104.0   Max.   :54.0
##      insulin      bmi      pedigree      age
##  Min.   : 0.00   Min.   :19.40   Min.   :0.0850   Min.   :21.00
## 1st Qu.: 0.00   1st Qu.:26.30   1st Qu.:0.2570   1st Qu.:24.00
## Median : 0.00   Median :31.60   Median :0.3910   Median :30.00
## Mean   : 63.77   Mean   :31.58   Mean   :0.4496   Mean   :33.31
## 3rd Qu.:105.00   3rd Qu.:36.00   3rd Qu.:0.5800   3rd Qu.:41.00
## Max.   :543.00   Max.   :50.00   Max.   :2.2880   Max.   :67.00
```

```
##      class      scored.class      scored.probability
## Min.    :0.0000   Min.    :0.0000   Min.    :0.02323
## 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.11702
## Median :0.0000   Median :0.0000   Median :0.23999
## Mean   :0.3149   Mean   :0.1768   Mean   :0.30373
## 3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.43093
## Max.    :1.0000   Max.    :1.0000   Max.    :0.94633
```

## 2. Raw Confusion Matrix

```
(confusion_matrix <- table("Actual"= df$class, "Predicted"=df$scored.class))
```

```
##      Predicted
## Actual    0    1
##      0 119    5
##      1   30   27
```

Here is the raw confusion matrix with rows reflecting Actual and columns are Predicted.

## Custom Metric Functions

### 3. Accuracy Function

The following function returns the accuracy of the predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

```
##' Return the accuracy of a given dataset. Note: the dataframe should have an observed class and predicted class
##'
##' @param df A dataframe
##' @param observed Column name holding the class
##' @param predicted Column name holding the predicted class
##' @examples
##' accuracy(myDF)
##' @return float
##' @export
accuracy <- function(df=NULL, observed='class', predicted='scored.class') {
  # Make sure a dataframe was passed and we have both class and predicted columns
  if (is.null(df) || !any(names(df)==observed) || !any(names(df) == predicted)) {
    return
  }

  # true negative, false negative, false positive, true positive
  cols = c("TN", "FN", "FP", "TP")
  confusion_matrix <- table("Actual" = df[[observed]],
                           "Predicted" = df[[predicted]])

  confusion_matrix <- data.frame(confusion_matrix, index = cols)
```

```

# calculate accuracy
accuracy_value <- (confusion_matrix$Freq[4] + confusion_matrix$Freq[1]) / sum(confusion_matrix$Freq)
return(accuracy_value)
}

# test function
(Accuracy <- accuracy(df))

```

```
## [1] 0.8066298
```

#### 4. Error Function

$$Error = \frac{FP + FN}{TP + FP + TN + FN}$$

The following function returns the classification error rate of the predictions.

```

error <- function(df) {
  # true negative, false negative, false positive, true positive
  cols = c("TN", "FN", "FP", "TP")
  confusion_matrix <- table("Actual"=df$class, "Predicted"=df$scored.class)
  confusion_matrix <- data.frame(confusion_matrix, index = cols)

  # calculate error
  error_value <- (confusion_matrix$Freq[2] + confusion_matrix$Freq[3]) / sum(confusion_matrix$Freq)
  return(error_value)
}

# test function
(Error <- error(df))

```

```
## [1] 0.1933702
```

We can verify the sum of the accuracy and error rates is equal to one.

```

# check sum
Accuracy + Error

```

```
## [1] 1
```

#### 5. Precision Function

$$Precision = \frac{TP}{TP + FP}$$

The following function returns the precision of the predictions.

```

precision <- function(df) {
  # true negative, false negative, false positive, true positive
  cols = c("TN", "FN", "FP", "TP")
  confusion_matrix <- table("Actual"=df$class, "Predicted"=df$scored.class)
  confusion_matrix <- data.frame(confusion_matrix, index = cols)

```

```

# calculate precision
error_value <- (confusion_matrix$Freq[4])/(confusion_matrix$Freq[4]+confusion_matrix$Freq[3])
return(error_value)
}

# test function
precision(df)

## [1] 0.84375

```

## 6. Sensitivity Function

$$Sensitivity = \frac{TP}{TP + FN}$$

The following function returns the sensitivity of the predictions.

```

sensitivity <- function(df) {
  # true negative, false negative, false positive, true positive
  cols = c("TN", "FN", "FP", "TP")
  confusion_matrix <- table("Actual"=df$class, "Predicted"=df$scored.class)
  confusion_matrix <- data.frame(confusion_matrix, index = cols)

  # calculate sensitivity
  error_value <- (confusion_matrix$Freq[4])/(confusion_matrix$Freq[4]+confusion_matrix$Freq[2])
  return(error_value)
}

# test function
(sensitivity(df))

## [1] 0.4736842

```

## 7. Specificity Function

$$Specificity = \frac{TN}{TN + FP}$$

The following function returns the specificity of the predictions.

```

specificity <- function(df) {
  # true negative, false negative, false positive, true positive
  cols = c("TN", "FN", "FP", "TP")
  confusion_matrix <- table("Actual"=df$class, "Predicted"=df$scored.class)
  confusion_matrix <- data.frame(confusion_matrix, index = cols)

  #calculate specificity
  error_value <- (confusion_matrix$Freq[1]) / (confusion_matrix$Freq[1] + confusion_matrix$Freq[3])
  return(error_value)
}

# test function
(specificity(df))

```

```
## [1] 0.9596774
```

## 8. F1 Score Function

$$F1Score = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

The following function returns the F1 score of the predictions.

```
f1_score <- function(df) {  
  # get precision and sensitivity from our custom functions  
  precision_value <- precision(df)  
  sensitivity_value <- sensitivity(df)  
  
  # calculate F1 Score  
  F1_Score = (2 * precision_value * sensitivity_value) / (precision_value + sensitivity_value)  
  return(F1_Score)  
}  
  
(f1_score(df))
```

```
## [1] 0.6067416
```

## 9. F1 Score Bounds

```
f1_function <- function(precision, sensitivity) {  
  f1score <- (2 * precision * sensitivity) / (precision + sensitivity)  
  return (f1score)  
}  
  
# 0 precision, 0.5 sensitivity  
(f1_function(0, .5))
```

```
## [1] 0
```

```
# 1 precision, 1 sensitivity  
(f1_function(1, 1))
```

```
## [1] 1
```

The F1 score is bounded from 0 to 1.

## 10. ROC Curve

```
roc_plot <- function(df, probability) {  
  set.seed(824)  
  
  x <- seq(0, 1, .01)
```

```

FPR <- numeric(length(x))
TPR <- FPR
pos <- sum(df$class == 1)
neg <- sum(df$class == 0)

for (i in 1:length(x)) {
  data_subset <- subset(df, df$scored.probability <= x[i])

  # true positive
  TP <- sum(data_subset[data_subset$class == 1, probability] > 0.5)

  # true negative
  TN <- sum(data_subset[data_subset$class == 0, probability] <= 0.5)

  # false positive
  FP <- sum(data_subset[data_subset$class == 0, probability] > 0.5)

  # false negative
  FN <- sum(data_subset[data_subset$class == 1, probability] <= 0.5)

  TPR[i] <- 1 - (TP + FN) / pos
  FPR[i] <- 1 - (TN + FP) / neg
}

classification_data <- data.frame(TPR, FPR)

ggplot <- ggplot(classification_data, aes(FPR, TPR))

plot = ggplot +
  geom_line() +
  geom_abline(intercept = 0) +
  ggtitle("ROC Curve for Classification data") +
  theme_bw()

height = (classification_data$TPR[-1] +
  classification_data$TPR[-length(classification_data$TPR)]) / 2
width = -diff(classification_data$FPR)
AUC = sum(height * width)

return (list(AUC = AUC, plot))
}

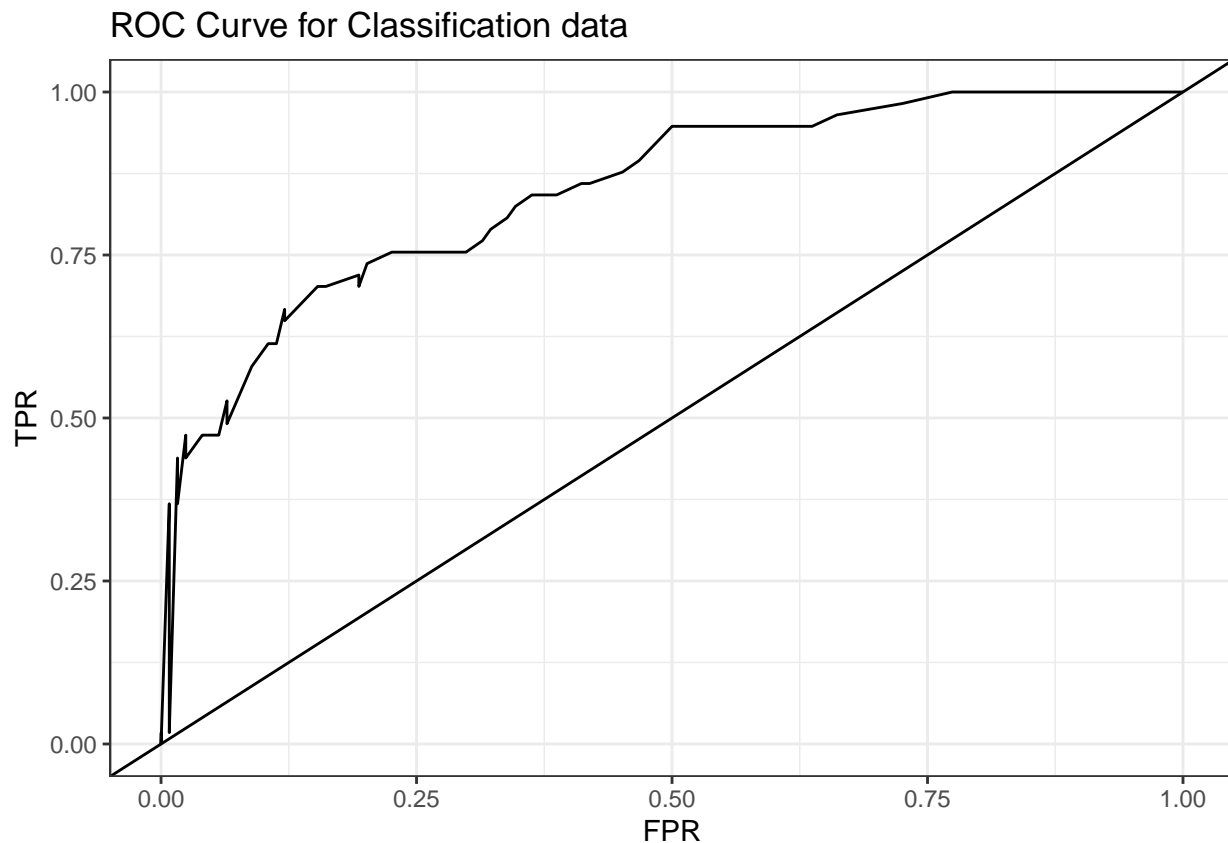
roc_plot(df, "scored.probability")

```

```

## $AUC
## [1] 0.8488964
##
## [[2]]

```



## 11. Use All Functions

```
Accuracy <- accuracy(df)
Error <- error(df)
Precision <- precision(df)
Sensitivity <- sensitivity(df)
Specificity <- specificity(df)
F1_score <- f1_score(df)
ROC <- roc_plot(df, "scored.probability")
AUC <- ROC$AUC

classification_data <- t(data.frame(Accuracy,
                                   Error,
                                   Precision,
                                   Sensitivity,
                                   Specificity,
                                   F1_score,
                                   AUC))

classification_data
```

```
##           [,1]
## Accuracy  0.8066298
## Error     0.1933702
## Precision  0.8437500
```

```
## Sensitivity 0.4736842
## Specificity 0.9596774
## F1_score    0.6067416
## AUC         0.8488964
```

## 12. Compare to caret Functions

```
cm <- confusionMatrix(data = as.factor(df$scored.class),
                      reference = as.factor(df$class),
                      positive = "1")
cm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 119  30
##              1   5  27
##
##              Accuracy : 0.8066
##              95% CI : (0.7415, 0.8615)
##              No Information Rate : 0.6851
##              P-Value [Acc > NIR] : 0.0001712
##
##              Kappa : 0.4916
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##              Sensitivity : 0.4737
##              Specificity : 0.9597
##              Pos Pred Value : 0.8438
##              Neg Pred Value : 0.7987
##              Prevalence : 0.3149
##              Detection Rate : 0.1492
##              Detection Prevalence : 0.1768
##              Balanced Accuracy : 0.7167
##
##              'Positive' Class : 1
##
```

```
Sensitivity == cm$byClass["Sensitivity"]
```

```
## Sensitivity
##          TRUE
```

```
Specificity == cm$byClass["Specificity"]
```

```
## Specificity
##          TRUE
```



```
Accuracy == cm$overall["Accuracy"]
```

```
## Accuracy  
## TRUE
```

Our homebrew R functions match with the `caret()` package functions.

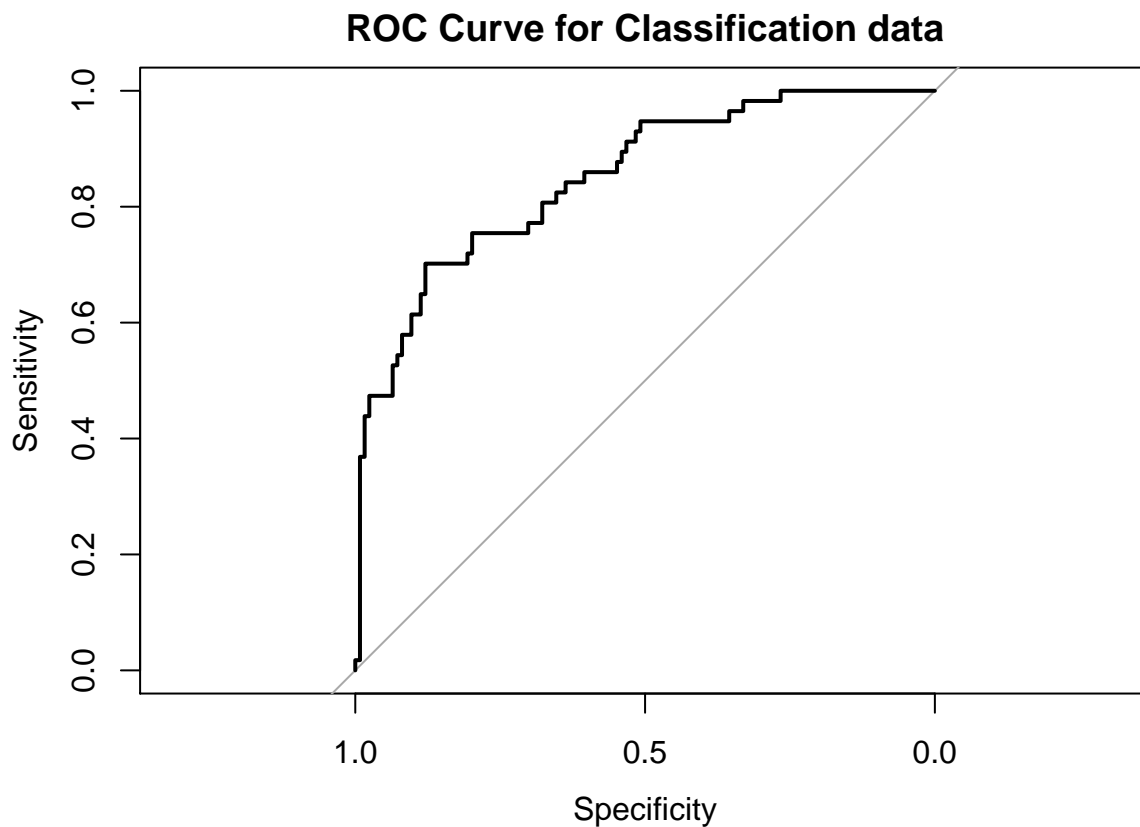
### 13. pROC

```
roc <- roc(df$class, df$scored.probability)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc, main="ROC Curve for Classification data")
```



```
# area under curve  
roc$auc
```

```
## Area under the curve: 0.8503
```

Our AUC was 0.8484 compared to pROC's 0.8503, which are within 0.2% of each other and thus convergence difference could be due to numerical integration under the curve.

## References

1. Kuhn and Johnson. Applied Predictive Modeling.
2. Web tutorials: [http://www.saedsayad.com/model\\_evaluation\\_c.htm](http://www.saedsayad.com/model_evaluation_c.htm)